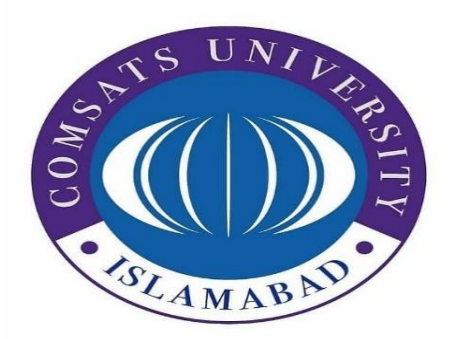


Lab Terminal

Name: Fizza Bukhari

Registration number: CIIT/SP21/BCS-007/ATK

Course: Compiler Construction

Submitted to: Sir Bilal Haider Bukhari

Date: 31 May 2024

Question #01:

Write an introduction of your compiler construction project

Introduction:

Compiler construction is a fundamental aspect of computer science that bridges the gap between high-level programming languages and machine code that can be executed by a computer's hardware. Our project, the Mini-Python Compiler, serves as an educational tool to demonstrate the essential phases of compiler construction, specifically for a subset of the Python programming language.

The Mini-Python Compiler project is designed to translate a simplified version of Python code into intermediate code and then into a hypothetical assembly-like language. This process involves several key stages, each critical for converting human-readable code into low-level instructions. The primary components of our compiler include lexical analysis, syntax analysis, semantic analysis, intermediate code generation, and target code generation.

For this mini-compiler, the following aspects of the Python language syntax have been covered

- Constructs like ‘if-else’ and ‘while’ and the required indentation for these loops.
- Nested loops
- Integer and float data types

Specific error messages are displayed based on the type of error. Syntax errors are handled using the yyerror() function, while the semantic errors are handled by making a call to a function that searches for a particular identifier in the symbol table. The line number is displayed as part of the error message.

As a part of error recovery, panic mode recovery has been implemented for the lexer. It recovers from errors in variable declaration. In case of identifiers, when the name begins with a digit, the compiler neglects the digit and considers the rest as the identifier name. Languages used to develop this project:

- C
- YACC
- LEX
- PYTHON

DIFFERENT MODULES OF PROJECT

Token And Symbol Table:

This folder contains the code that outputs the tokens and the symbol table.

Abstract Syntax Tree:

This folder contains the code that displays the abstract syntax tree.

Intermediate Code Generation:

This folder contains the code that generates the symbol table before optimisations and the intermediate code.

Optimized ICG:

This folder contains the code that generates the symbol table after optimizations, the quadruples table and the optimized intermediate code.

Target Code:

This folder contains the code that displays the assembly code/target code

Different Files:

proj.l:

It is the Lexical analyser file which defines all the terminals of the productions stated in the yacc file. It contains regular expressions.

proj1.y:

Yacc file is where the productions for the conditional statements like if-else and while and expressions are mentioned. This file also contains the semantic rules defined against every production necessary. Rules for producing three address code is also present.

final.py:

It is the python file which converts the ICG to target code using regex.

inp.py:

The input python code which will be parsed and checked for semantic correctness by executing the lex and yacc files along with it.

Question #02:

Give a sample input and output for your compiler construction project

Sample input:

```
1   a=10
2   b=9
3   c=a+b+100
4   e=10
5   f=8
6   d=e*f
7   if(a>=b):
8       a=a+b
9       g=e*f*100
10
11  u=10
12  j=99
```

Output:

```
MOV R0, #10
MOV R1, #9
MOV R2, #119
MOV R3, #8
MOV R4, #80
l0:
MOV R5, #0
BNEZ R5, l1
MOV R6, #19
MOV R7, #8000
l1:
MOV R8, #99
ST b, R1
ST c, R2
ST e, R0
ST f, R3
ST d, R4
ST a, R6
ST g, R7
ST u, R0
ST j, R8
```

How the process being:

Input:

```
1   a=10
2   b=9
3   c=a+b+100
4   e=10
5   f=8
6   d=e*f
7   if(a>=b):
8       a=a+b
9       g=e*f*100
10
11  u=10
12  j=99
```

Tokens and Symbol Table:

```
ID equal int
ID equal int
ID equal ID plus ID plus int
ID equal int
ID equal int
ID equal ID mul ID
if special_start ID greaterthanequal ID special_end colon
indent ID equal ID plus ID
indent ID equal ID mul ID mul int

ID equal int
ID equal int
------------------------------PARSE SUCCESSFUL--------------------------

------------------SYMBOL TABLE----------------
--------------------------------------------
LABEL   TYPE          VALUE   SCOPE   LINENO
a       IDENTIFIER    19      local   8
b       IDENTIFIER    9       global  2
c       IDENTIFIER    119     global  3
e       IDENTIFIER    10      global  4
f       IDENTIFIER    8       global  5
d       IDENTIFIER    80      global  6
g       IDENTIFIER    8000    local   9
u       IDENTIFIER    10      global  11
j       IDENTIFIER    99      global  12
```

Abstract Syntax Tree:

```
-----------Abstract Syntax Tree-------------
( SEQ ( =  a  10 )( SEQ ( =  b  9 )( SEQ ( =  c ( + ( +  a  b ) 100 ))( SEQ ( =  e  10 )( SEQ ( =  f  8 )( SEQ ( =  d
( *  e  f ))( SEQ ( IF ( >=  a  b )( SEQ ( =  a ( +  a  b ))( SEQ ( =  g ( * ( *  e  f ) 100 )) NULL )))( SEQ ( =  u
10 )( SEQ ( =  j  99 ) NULL )))))))))
```

Symbol Table and Unoptimized Intermediate Code:

```
-----------------SYMBOL TABLE before Optimisations---------------
-----------------------------------------------------------------
LABEL   TYPE            VALUE   SCOPE   LINENO
a       identifier      9       local   8
b       identifier      9       global  2
t0      identifier      19      -       2
t1      identifier      119     -       3
c       identifier      119     global  3
e       identifier      10      global  4
f       identifier      8       global  5
t2      identifier      80      -       6
d       identifier      80      global  6
t3      identifier      0       -       6
t4      identifier      9       -       8
t5      identifier      80      -       8
t6      identifier      8000    -       9
g       identifier      8000    local   9
u       identifier      10      local   11
j       identifier      99      local   12

-------------ICG without optimisation-------------
a=10
b=9
t0=a+b
t1=t0+100
c=t1
e=10
f=8
t2=e*f
d=t2
l0 : t3=a>=b
if not t3 goto l1
t4=a+b
a=t4
t5=e*f
t6=t5*100
g=t6
l1 : u=10
j=99
```

Symbol Table, Quadruples Table and Optimized Intermediate Code:

```
-----------------SYMBOL TABLE after Optimisations---------------
------------------------------------------------------------------
LABEL   TYPE            VALUE   SCOPE   LINENO
a       identifier      19      local   8
b       identifier      9       global  2
t0      identifier      19      -       2
t1      identifier      119     -       3
c       identifier      119     global  3
e       identifier      10      global  4
f       identifier      8       global  5
t2      identifier      80      -       5
d       identifier      80      global  6
t3      identifier      1       -       6
t4      identifier      0       -       6
t5      identifier      8000    -       8
g       identifier      8000    local   9
u       identifier      10      local   11
j       identifier      99      local   12

------------QUADRUPLES--------------
        op      arg1    arg2    result
        =       10              a
        =       9               b
        +       a       b       t0
        +       t0      100     t1
        =       t1              c
        =       10              e
        =       8               f
        *       e       f       t2
        =       t2              d
        Label                   l0
        >=      a       b       t3
        goto                    l1
        =       t0              a
        *       t2      100     t5
        =       t5              g
        Label                   l1
        =       10              u
        =       99              j
```

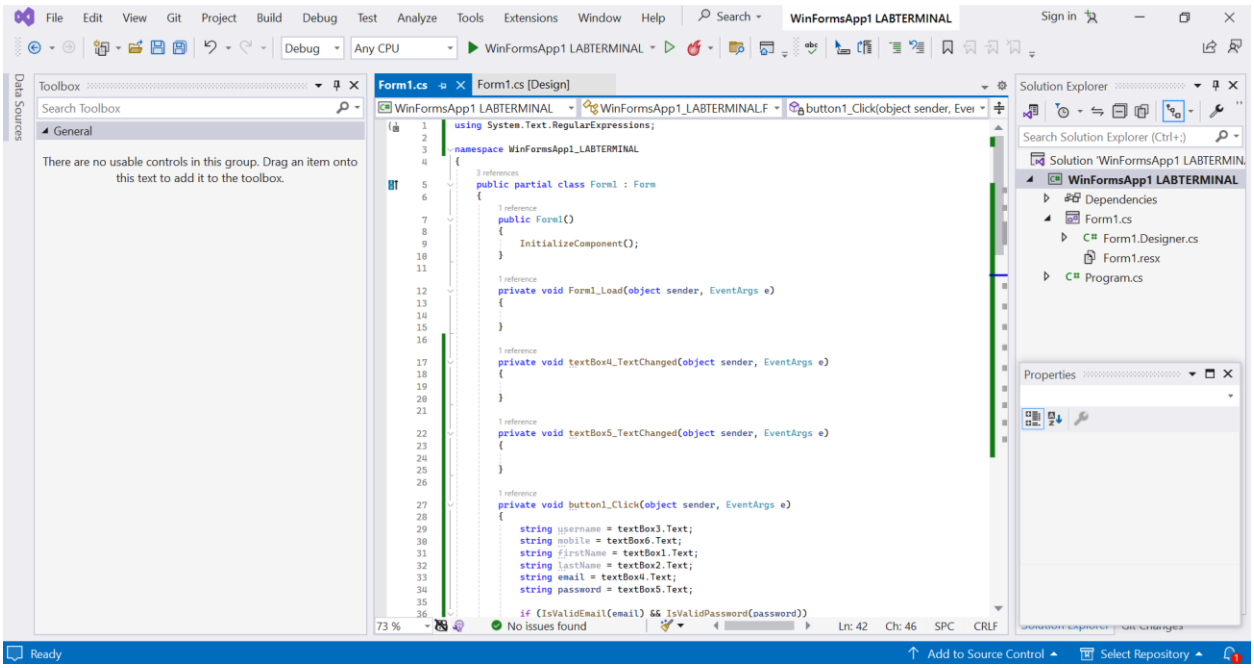
```
ICG with optimisations(Packing temporaries & Constant Propagation)
a = 10
b = 9
t0 = 10 + 9
t1 = 19 + 100
c = 119
e = 10
f = 8
t2 = 10 * 8
d = 80
l0:
t3 = 10 >= 9
t4 = not 1
if 0 goto l1
a = 19
t5 = 80 * 100
g = 8000
l1:
u = 10
j = 99
```

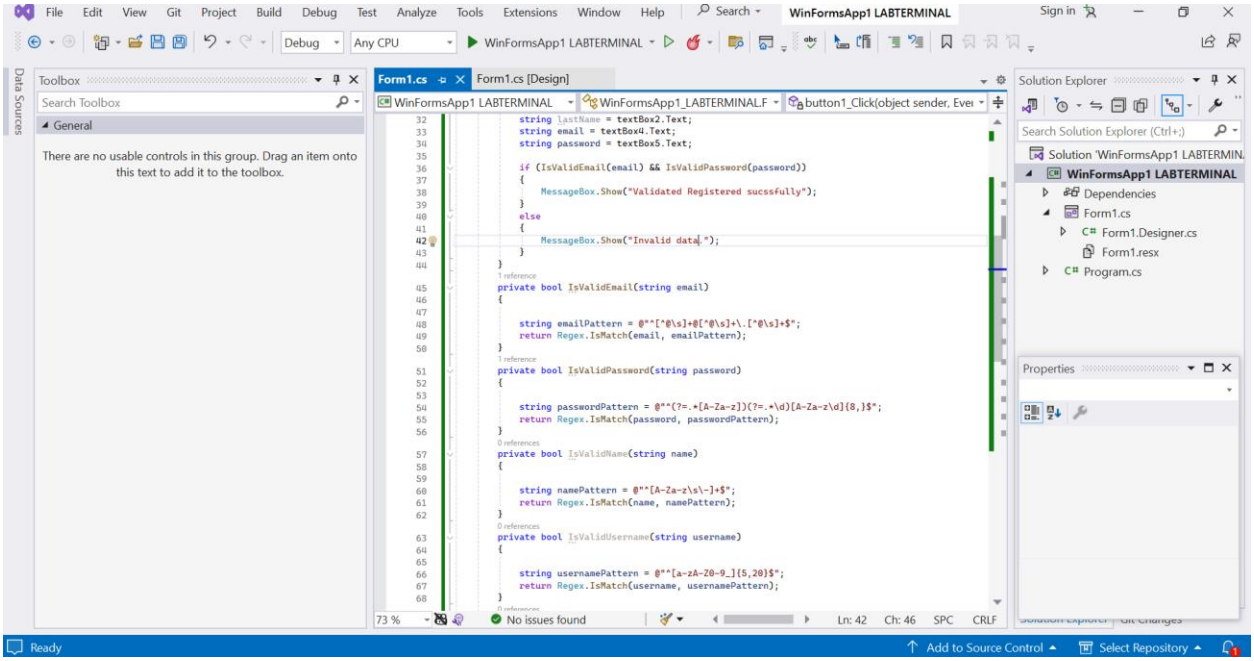
Target Code:

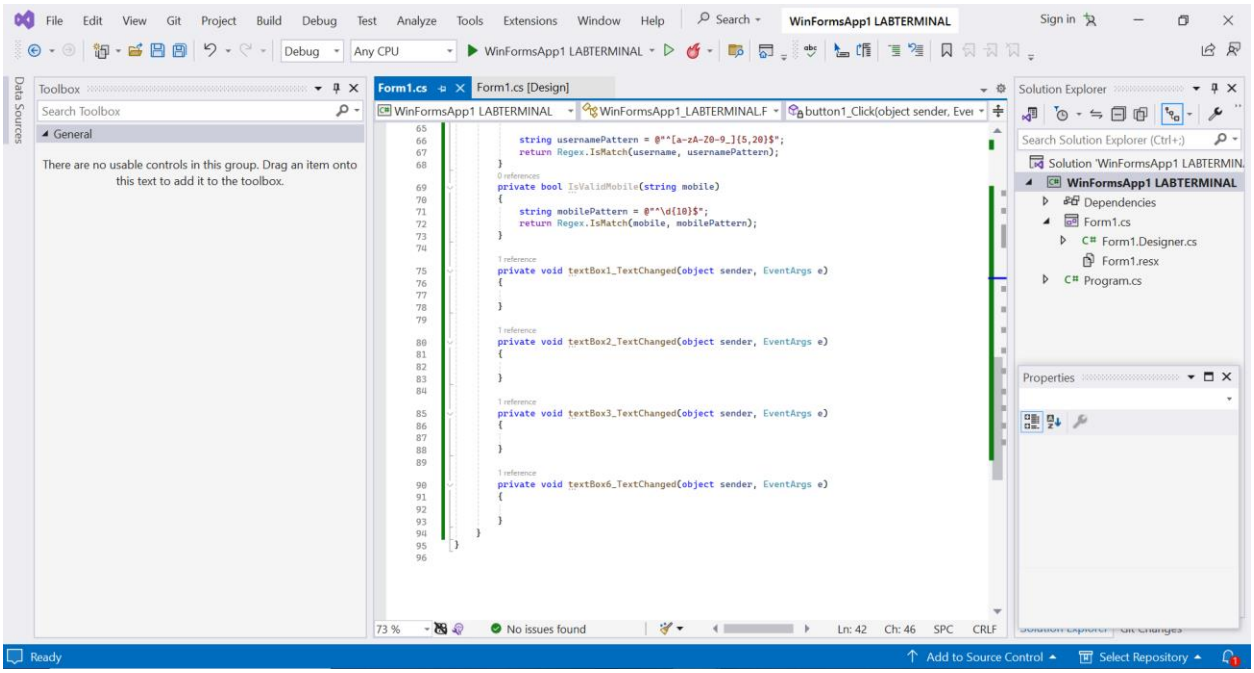
```
MOV R0, #10
MOV R1, #9
MOV R2, #119
MOV R3, #8
MOV R4, #80
l0:
MOV R5, #0
BNEZ R5, l1
MOV R6, #19
MOV R7, #8000
l1:
MOV R8, #99
ST b, R1
ST c, R2
ST e, R0
ST f, R3
ST d, R4
ST a, R6
ST g, R7
ST u, R0
ST j, R8
```

Question #03:

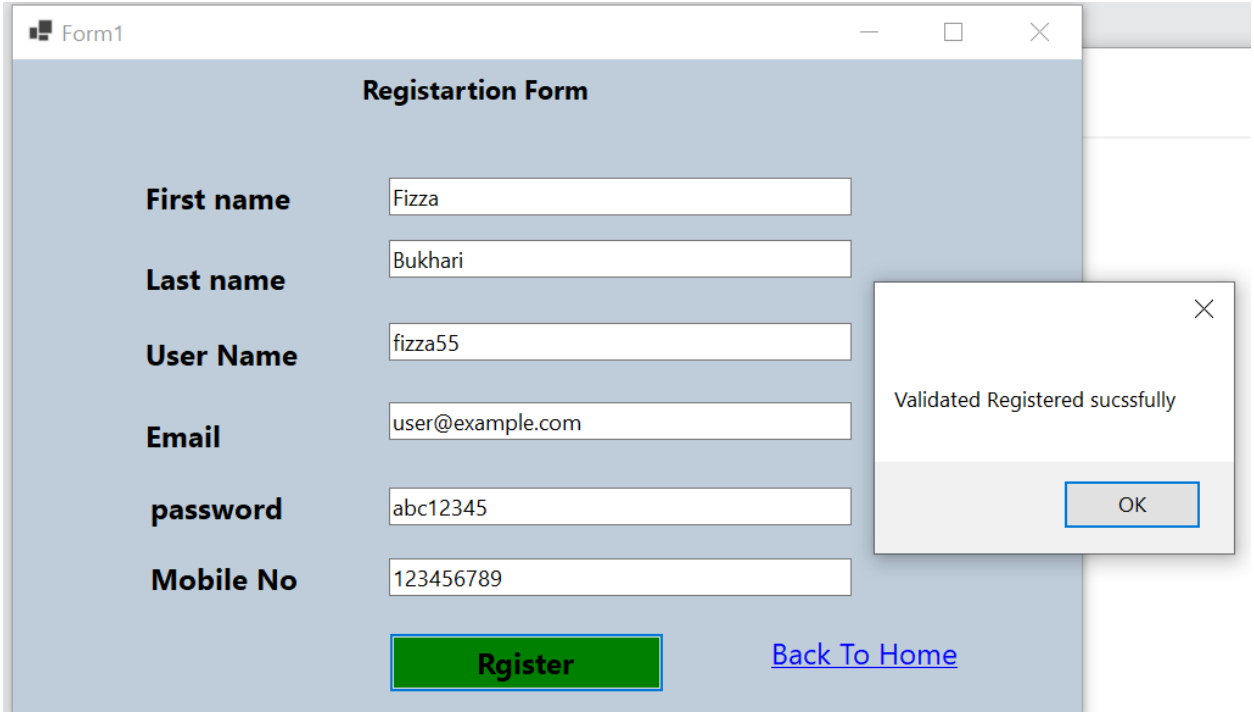
Code:





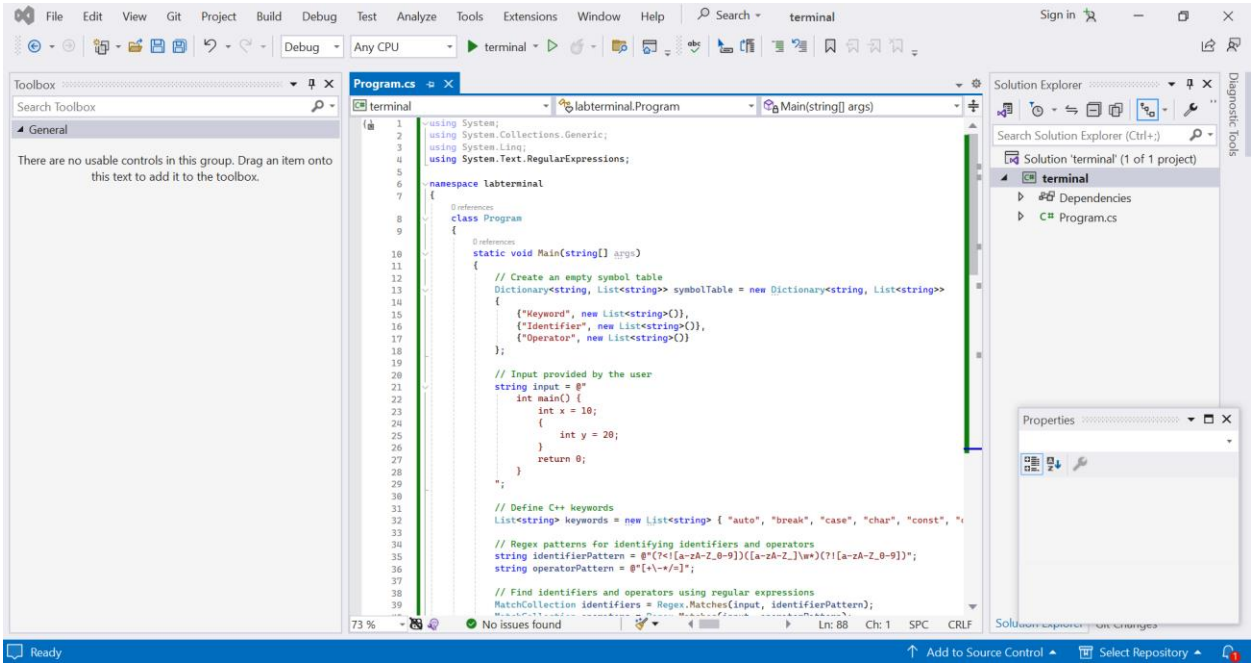

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
WinFormsApp1 LABTERMINAL
Toolbox
Search Toolbox
General
There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.
Form1.cs
Form1.cs [Design]
Solution Explorer
Search Solution Explorer (Ctrl+;)
WinFormsApp1 LABTERMINAL
Dependencies
Form1.cs
Form1.Designer.cs
Form1.resx
Program.cs
Properties
No issues found
Ln: 42 Ch: 46 SPC CRLF
Ready
Add to Source Control
Select Repository

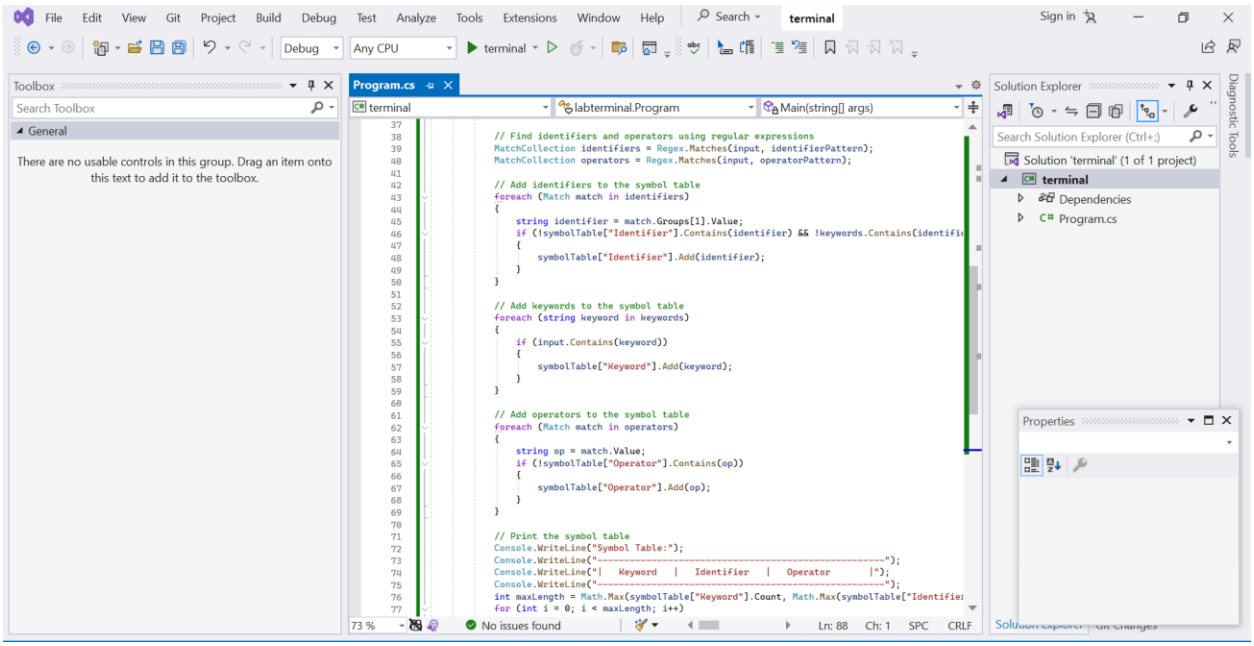
Output:

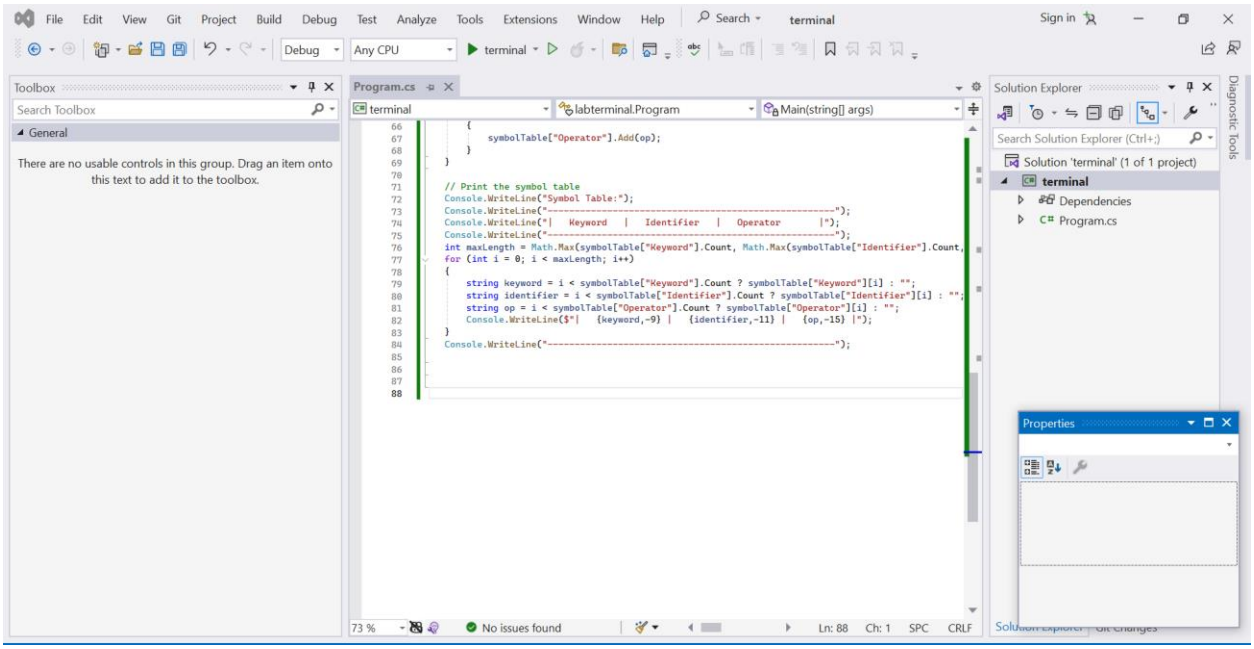

Form1
Registartion Form
First name
Fizza
Last name
Bukhari
User Name
fizza55
Email
user@example.com
password
abc12345
Mobile No
123456789
Rgister
Back To Home
Validated Registered sucssfully
OK

Question#04:

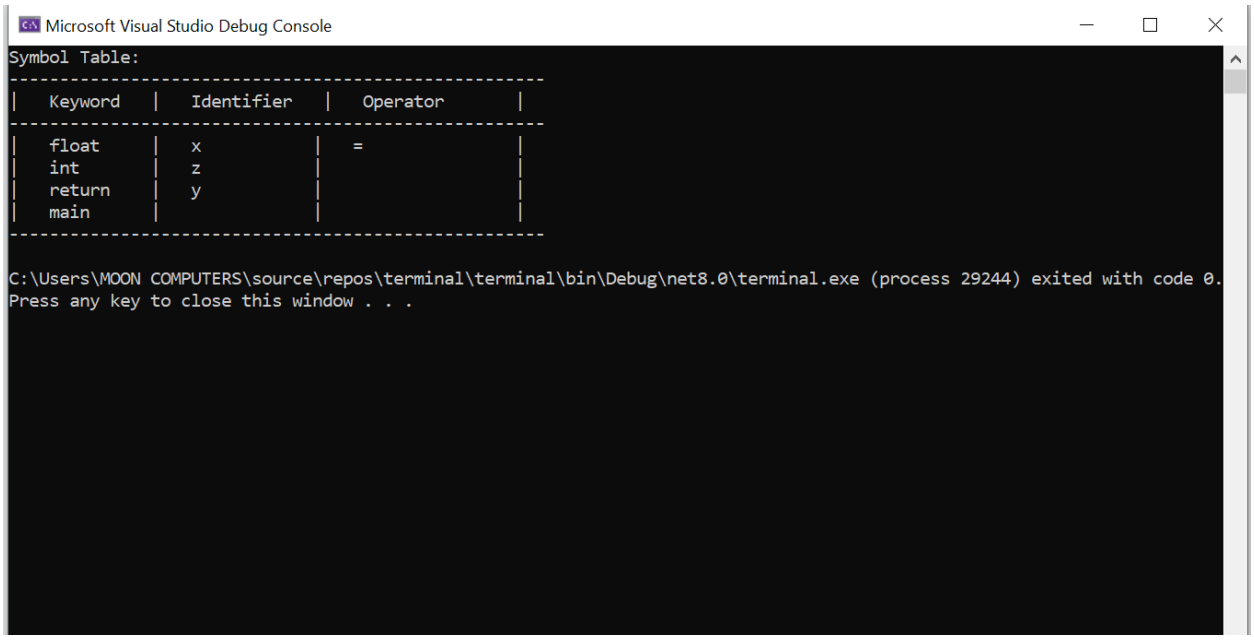
Code:






File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Search terminal
Sign in
Debug
Any CPU
terminal
Toolbox
Search Toolbox
General
There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.
Program.cs
terminal
labterminal.Program
Main(string[] args)
symbolTable["Operator"].Add(op);
// Print the symbol table
Console.WriteLine("Symbol Table:");
Console.WriteLine("----------");
Console.WriteLine("| Keyword | Identifier | Operator |");
Console.WriteLine("----------");
int maxLength = Math.Max(symbolTable["Keyword"].Count, Math.Max(symbolTable["Identifier"].Count,
for (int i = 0; i < maxLength; i++)
string keyword = i < symbolTable["Keyword"].Count ? symbolTable["Keyword"][i] : "";
string identifier = i < symbolTable["Identifier"].Count ? symbolTable["Identifier"][i] : "";
string op = i < symbolTable["Operator"].Count ? symbolTable["Operator"][i] : "";
Console.WriteLine($"| {keyword,-9} | {identifier,-11} | {op,-15} |");
Console.WriteLine("----------");
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'terminal' (1 of 1 project)
terminal
Dependencies
Program.cs
Properties
Diagnostic Tools
73 %
No issues found
Ln: 88 Ch: 1 SPC CRLF

Output:


Microsoft Visual Studio Debug Console
Symbol Table:
Keyword	Identifier	Operator
float	x	=
int	z	
return	y	
main		
C:\Users\MOON COMPUTERS\source\repos\terminal\terminal\bin\Debug\net8.0\terminal.exe (process 29244) exited with code 0.
Press any key to close this window . . .